Kinematics & Dynamics

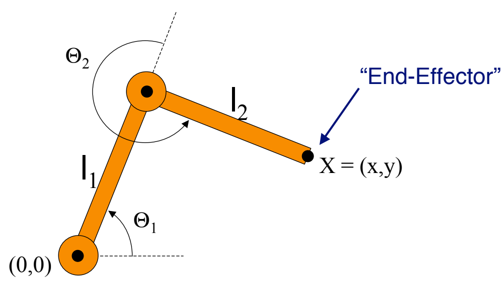
Adam Finkelstein
Princeton University
COS 426, Spring 2005

Overview

- Kinematics
  - Considers only motion
  - Determined by positions, velocities, accelerations
- Dynamics
  - Considers underlying forces
  - Compute motion from initial conditions and physics

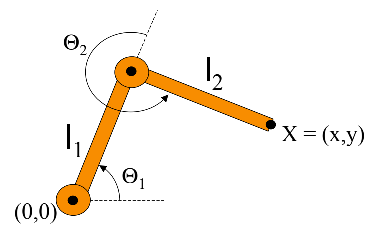
Example: 2-Link Structure

- Two links connected by rotational joints



Forward Kinematics

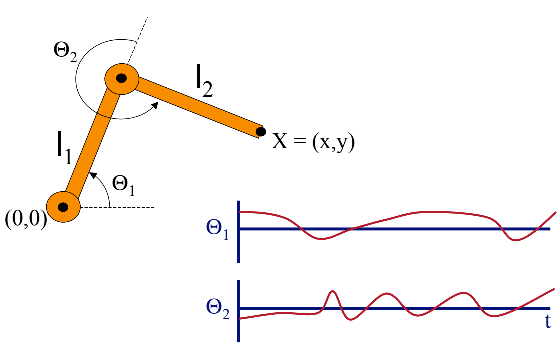
- Animator specifies joint angles: Θ_1 and Θ_2
- Computer finds positions of end-effector: X



$$X = (l_1 \cos\Theta_1 + l_2 \cos(\Theta_1 + \Theta_2), l_1 \sin\Theta_1 + l_2 \sin(\Theta_1 + \Theta_2))$$

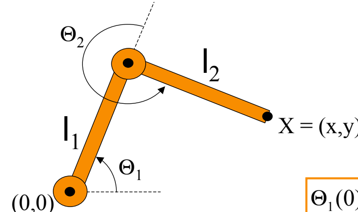
Forward Kinematics

- Joint motions can be specified by spline curves



Forward Kinematics

- Joint motions can be specified by initial conditions and velocities

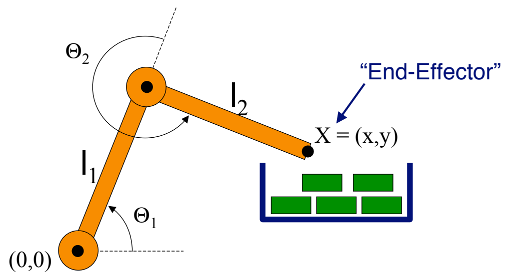


$$\Theta_1(0) = 60^\circ \qquad \Theta_2(0) = 250^\circ$$

$$\frac{d\Theta_1}{dt} = 1.2 \qquad \frac{d\Theta_2}{dt} = -0.1$$

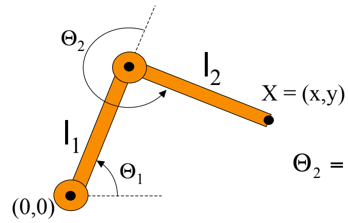
Example: 2-Link Structure

- What if animator knows position of "end-effector"



Inverse Kinematics

- Animator specifies end-effector positions: X
- Computer finds joint angles: Θ_1 and Θ_2:

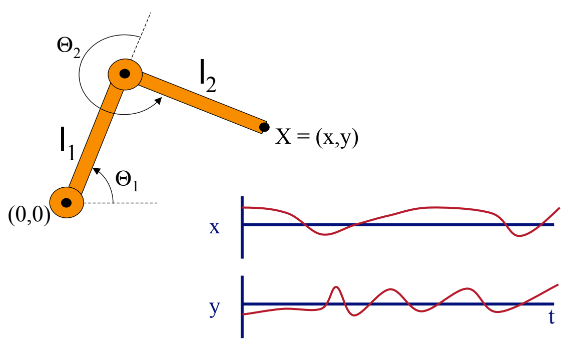


$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2) x + (l_1 + l_2 \cos(\Theta_2)) y}{(l_2 \sin(\Theta_2)) y + (l_1 + l_2 \cos(\Theta_2)) x}$$

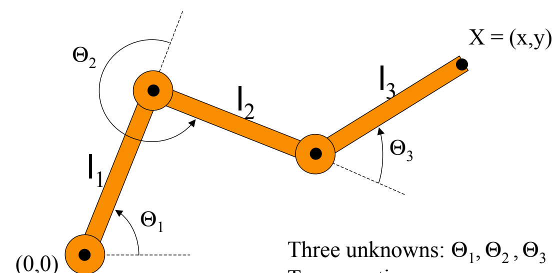
Inverse Kinematics

- End-effector postions specified by spline curves



Inverse Kinematics

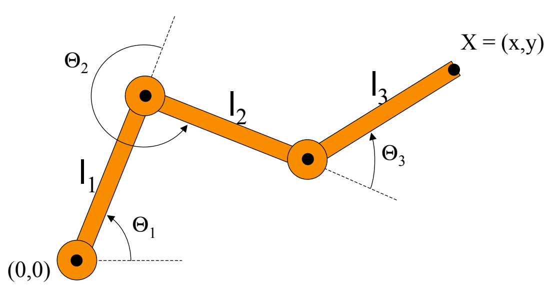
- Problem for more complex structures
  - System of equations is usually under-defined
  - Multiple solutions



Three unknowns: $\Theta_1, \Theta_2, \Theta_3$
Two equations: x, y

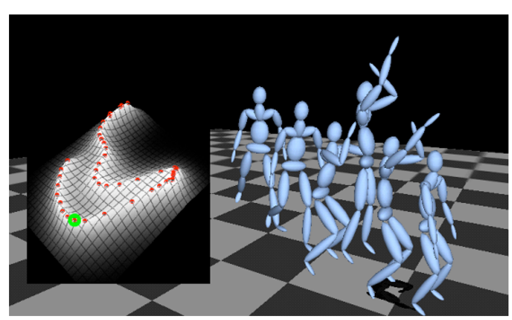
Inverse Kinematics

- Solution for more complex structures:
  - Find best solution (e.g., minimize energy in motion)
  - Non-linear optimization



Inverse Kinematics

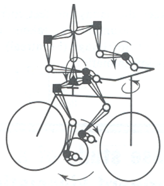
- Style-based IK: optimize for learned style



Growchow 04

Summary of Kinematics

- Forward kinematics
  - Specify conditions (joint angles)
  - Compute positions of end-effectors
- Inverse kinematics
  - "Goal-directed" motion
  - Specify goal positions of end effectors
  - Compute conditions required to achieve goals



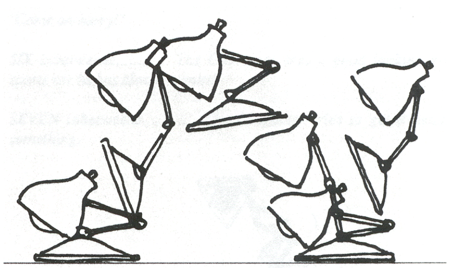
Inverse kinematics provides easier specification for many animation tasks, but it is computationally more difficult

Overview

- Kinematics
  - Considers only motion
  - Determined by positions, velocities, accelerations
- Dynamics
  - Considers underlying forces
  - Compute motion from initial conditions and physics
  - Active dynamics: objects have muscles or motors
  - Passive dynamics: external forces only

Dynamics

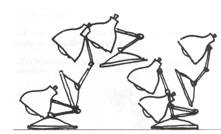
- Simulation of physics insures realism of motion



Lasseter `87

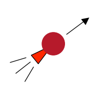
Spacetime Constraints

- Animator specifies constraints:
  - What the character's physical structure is
    - » e.g., articulated figure
  - What the character has to do
    - » e.g., jump from here to there within time t
  - What other physical structures are present
    - » e.g., floor to push off and land
  - How the motion should be performed
    - » e.g., minimize energy



Spacetime Constraints

- Computer finds the "best" physical motion satisfying constraints
- Example: particle with jet propulsion
  - **x**(t) is position of particle at time t
  - **f**(t) is force of jet propulsion at time t
  - Particle's equation of motion is:



$$mx'' - f - mg = 0$$

  - Suppose we want to move from a to b within t_0 to t_1 with minimum jet fuel:

$$\text{Minimize} \int_{t_0}^{t_1} |f(t)^2| dt \quad \text{subject to } x(t_0)=a \text{ and } x(t_1)=b$$

Witkin & Kass `88

Spacetime Constraints

- Discretize time steps:

$$x'_i = \frac{x_i - x_{i-1}}{h}$$

$$x''_i = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}$$

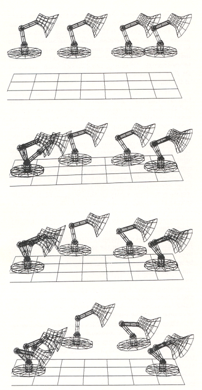
$$m\left(x''_i = \frac{x_{i+1} - 2x_i + x_{i-1}}{h^2}\right) - f_i - mg = 0$$

$$\text{Minimize } h \sum_i |f_i|^2 \quad \text{subject to } x_0=a \text{ and } x_1=b$$

Witkin & Kass `88

Spacetime Constraints

- Solve with iterative optimization methods



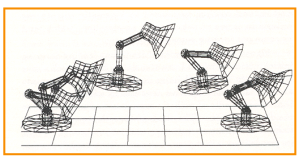
Witkin & Kass `88

Spacetime Constraints

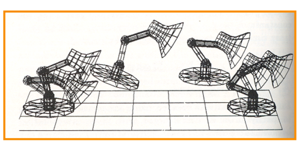
- Advantages:
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints
- Challenges:
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization

Spacetime Constraints

- Adapting motion:



Original Jump

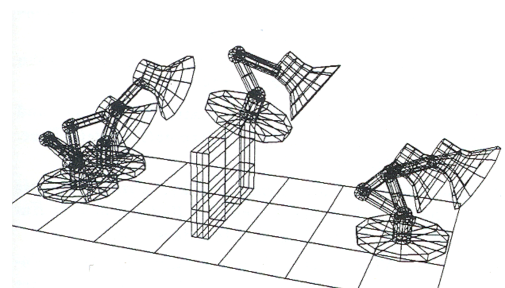


Heavier Base

Witkin & Kass `88

Spacetime Constraints

- Adapting motion:

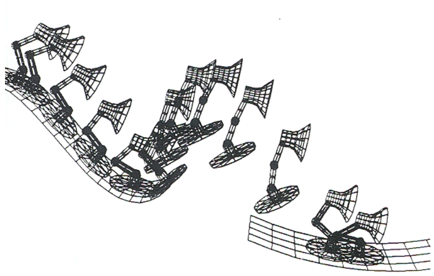


Hurdle

Witkin & Kass `88

Spacetime Constraints

- Adapting motion:



Ski Jump

Witkin & Kass `88

Motion Sketching

- Plausible motion matches sketched constraints

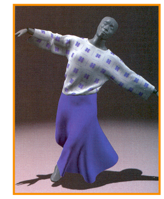
Popovic 03

Spacetime Constraints

- Advantages:
  - Free animator from having to specify details of physically realistic motion with spline curves
  - Easy to vary motions due to new parameters and/or new constraints
- Challenges:
  - Specifying constraints and objective functions
  - Avoiding local minima during optimization

Passive Dynamics

- Other physical simulations:
  - Rigid bodies
  - Soft bodies
  - Cloth
  - Liquids
  - Gases
  - etc.

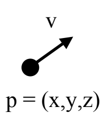


Cloth
(Baraff & Witkin `98)

Hot Gases
(Foster & Metaxas `97)

Particle Systems

- A particle is a point mass
  - Mass
  - Position
  - Velocity
  - Acceleration
  - Color
  - Lifetime



- Use lots of particles to model complex phenomena
  - Keep array of particles

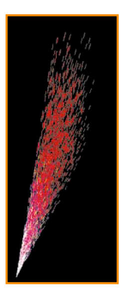
Particle Systems

- For each frame:
  - Create new particles and assign attributes
  - Delete any expired particles
  - Update particles based on attributes and physics
  - Render particles

Creating/Deleting Particles

- Where to create particles?
  - Around some center
  - Along some path
  - Surface of shape
  - Where particle density is low
- When to delete particles?
  - Where particle density is high
  - Life span
  - Random

This is where user controls animation



Example: Wrath of Khan

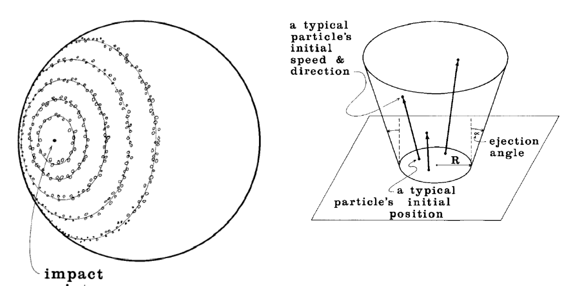
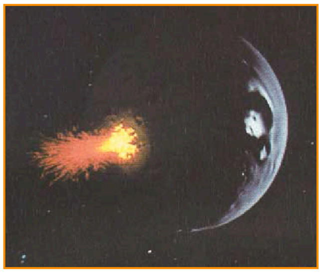


Fig. 2. Distribution of particle systems on the planet's surface.

Reeves

Example: Wrath of Khan



Reeves

Example: Wrath of Khan

Fig. 7. Wall of fire about to engulf camera.

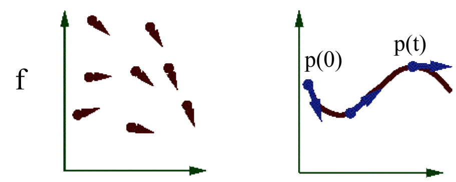
Reeves

Equations of Motion

- Newton's Law for a point mass
  - o $f = ma$
- Update every particle for each time step
  - o $a(t+\Delta t) = g$
  - o $v(t+\Delta t) = v(t) + a(t)*\Delta t$
  - o $p(t+\Delta t) = p(t) + v(t)*\Delta t + a(t)^2*\Delta t/2$

Solving the Equations of Motion

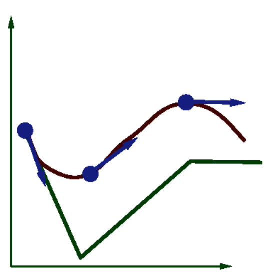
- Initial value problem
  - o Know p(0), v(0), a(0)
  - o Can compute force at any time and position
  - o Compute p(t) by forward integration



Hodgins

Solving the Equations of Motion

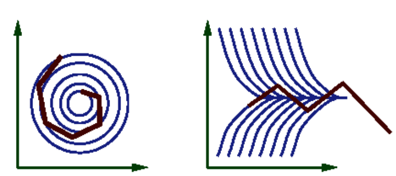
- Euler integration
  - o $p(t+\Delta t)=p(t) + \Delta t\, f(x,t)$



Hodgins

Solving the Equations of Motion

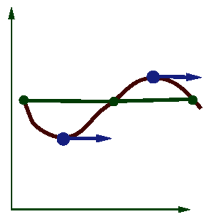
- Euler integration
  - o $p(t+\Delta t)=p(t) + \Delta t\, f(x,t)$
- Problem:
  - o Accuracy decreases as Δt gets bigger



Hodgins

Solving the Equations of Motion

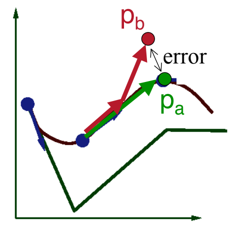
- Midpoint method (2nd order Runge-Kutta)
  - o Compute an Euler step
  - o Evalute f at the midpoint
  - o Take an Euler step using midpoint force
    - » $p(t+\Delta t)=p(t) + \Delta t\ f(\ p(t) + 0.5*\Delta t\ f(t),t)$



Hodgins

Solving the Equations of Motion

- Adapting step size
  - o Compute p_a by taking one step of size h
  - o Compute p_b by taking 2 steps of size h/2
  - o Error = $|\ p_a\ - p_b\ |$
  - o Adjust step size by factor (epsilon/error)$^{1/f}$

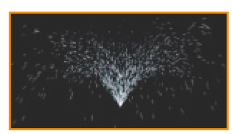


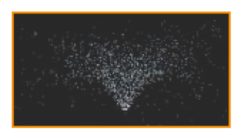
Particle System Forces

- Force fields
  - o Gravity, wind, pressure
- Viscosity/damping
  - o Liquids, drag
- Collisions
  - o Environment
  - o Other particles
- Other particles
  - o Springs between neighboring particles (mesh)
  - o Useful for cloth

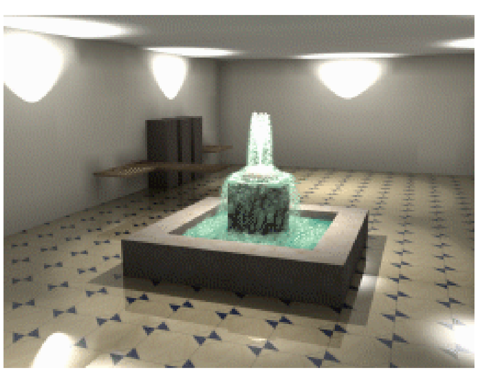
Rendering Particles

- Volumes
  - o Ray casting, etc.
- Points
  - o Render as individual points
- Line segments
  - o Motion blur over time





Example: Fountain

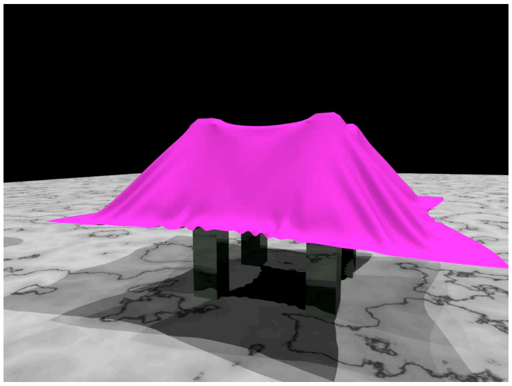


Particle System API

More Passive Dynamics Examples

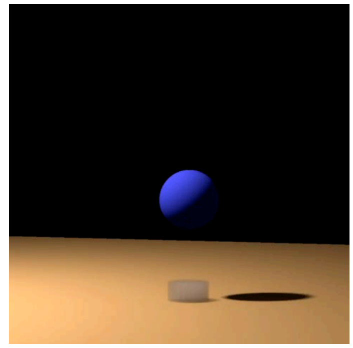
- Spring meshes
- Level sets
- Collisions
- etc.

Example: Cloth



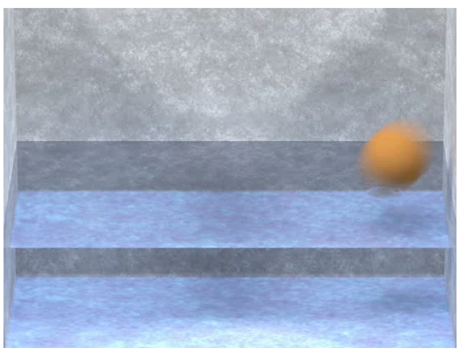
Fedkiw

Example: Smoke



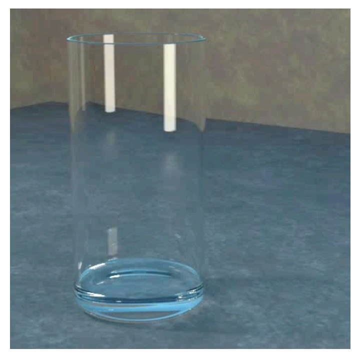
Fedkiw

Example: Water



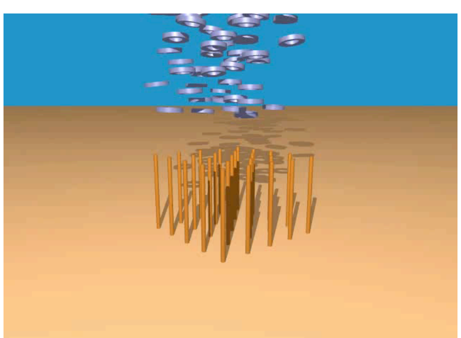
Fedkiw

Example: Water



Fedkiw

Example: Rigid Body Contact



Fedkiw

Summary

- Kinematics
  - o Forward kinematics
    - » Animator specifies joints (hard)
    - » Compute end-effectors (easy - assn 4!)
  - o Inverse kinematics
    - » Animator specifies end-effectors (easier)
    - » Solve for joints (harder)
- Dynamics
  - o Space-time constraints
    - » Animator specifies structures & constraints (easiest)
    - » Solve for motion (hardest)
  - o Also other physical simulations